Image Super Resolution using GAN (SRGAN)

Aravind Guhan A - 510517042

Ashwani Kumar Dubey - 510517064

Sankha Subhra Mandal - 510517063

Sajal Soni - 510517061

Supervisor : Prof. Jaya Sil

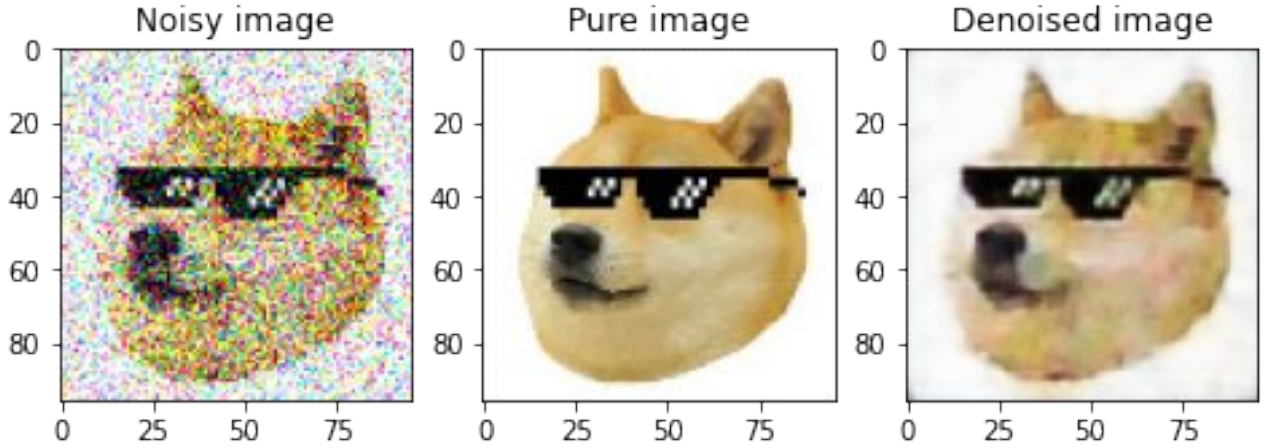
What we have done previously

- We have Implemented Image Denoising using Convolutional Neural Network.
- We have Implemented it on handwritten digits from MNIST dataset and STL10 dataset
- Step by step Implementation of the model:

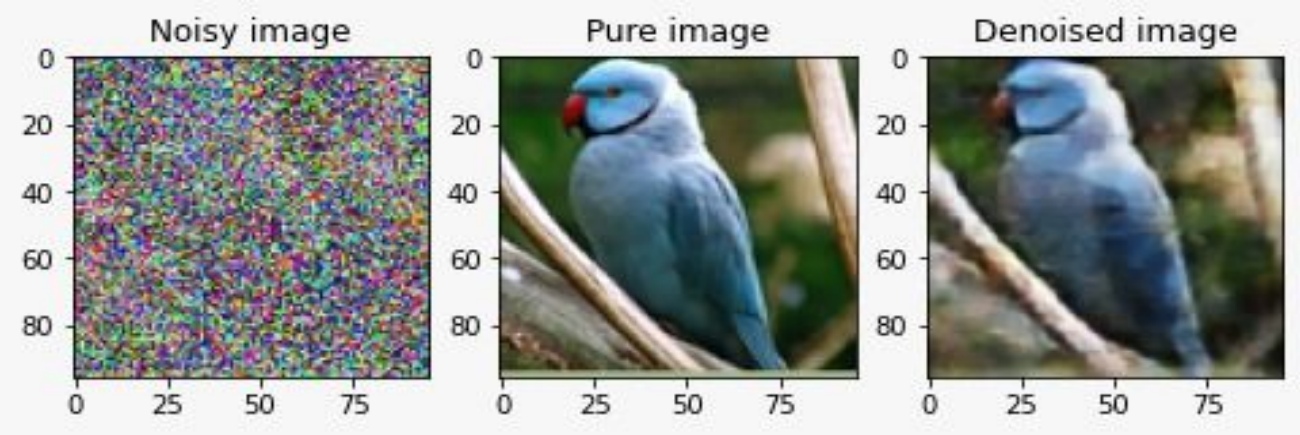
1. Load, reshape, scale and add noise to data.
2. Train CNN on noise merged training data.
3. Get Denoised Data that should Replicate the Original Data.
4. Compare its performance with the Original Data.

Results after training STL-10 Dataset

The takeaway from this model is that, the results are noiseless but it degrades in image quality.



Results after training STL-10 Dataset



Problem Statement

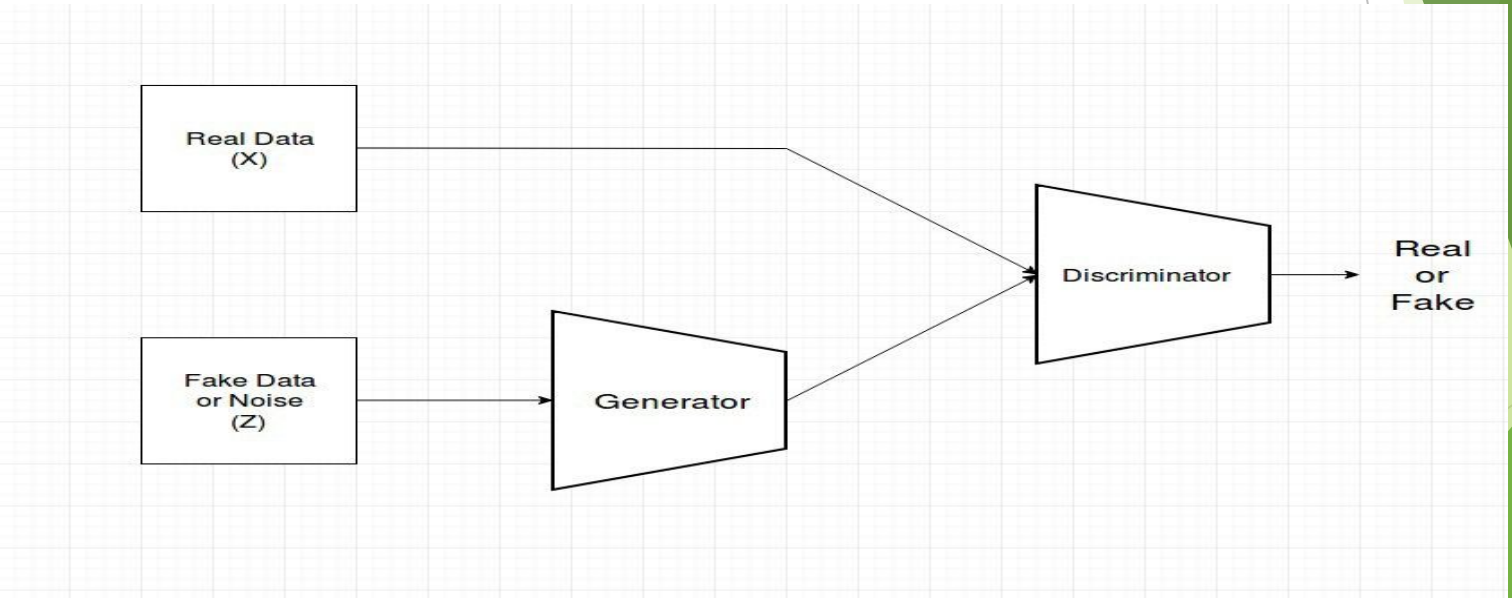
- Hence by seeing the results we can say that Constructing a Denoising Autoencoder for Images with MSE in the Final Layer outputs Denoised Blurry Images.
- So Now we are using Generative Adversarial Networks (GAN) to improve the images (Resolution and High Frequency Areas).
- To be more specific we are using Super Resolution Generative Adversarial Networks to Deblur the Image.
- We have used dataset of cat's image to train our model.

GAN (Generative Adversarial Networks)

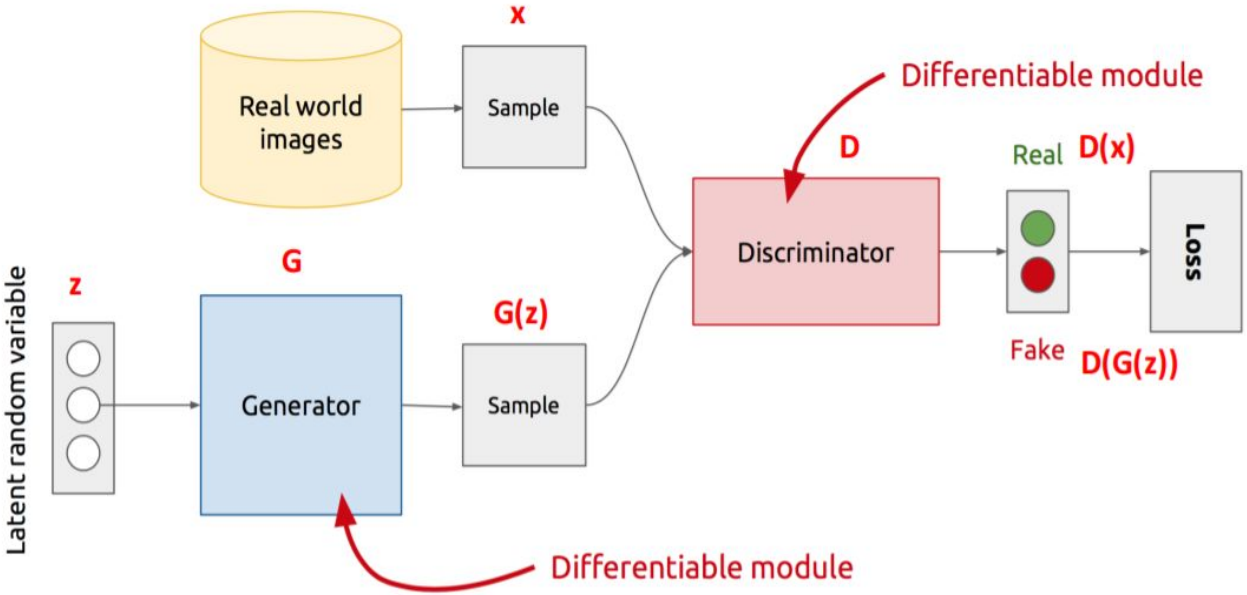
To understand GANs, first we need to understand what a generative model is.

- In machine learning, the two main classes of models are generative and discriminative.
- A discriminative model is one that discriminates between two (or more) different classes of data.
- A generative model on the other hand doesn't know anything about classes of data. Instead, its purpose is to generate new data which fits the distribution of the training data.

- GANs consist of a Generator and Discriminator

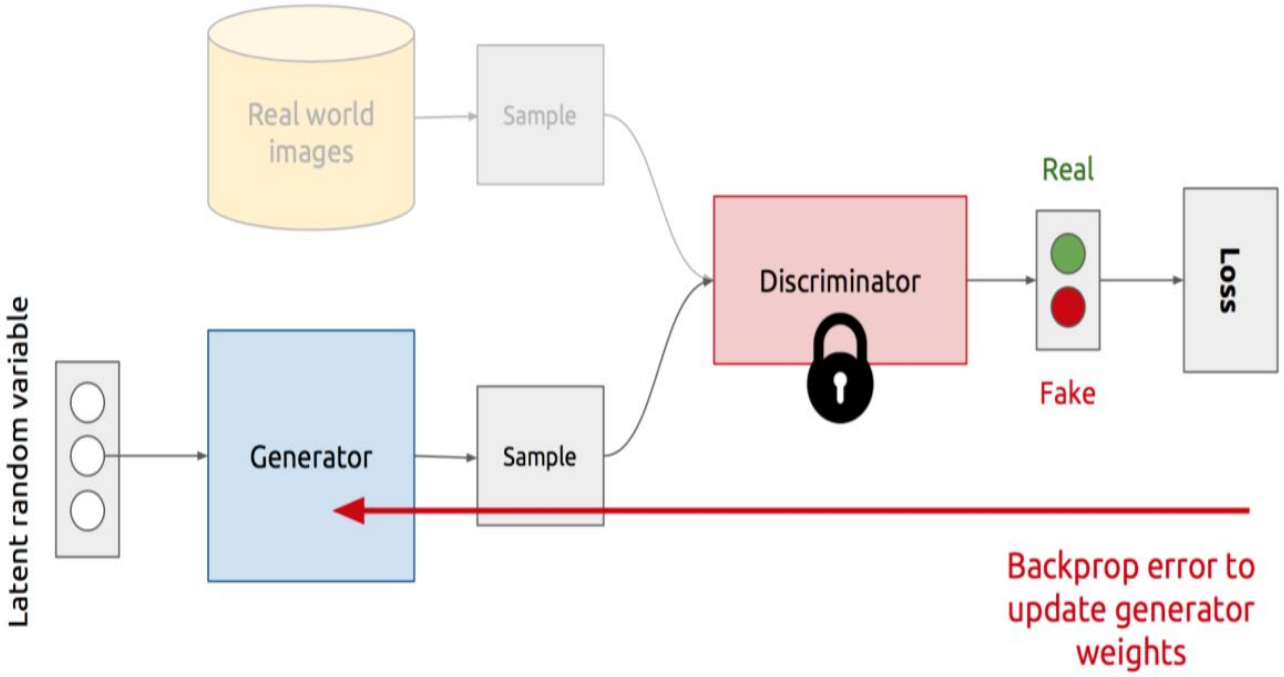


GAN's Architecture

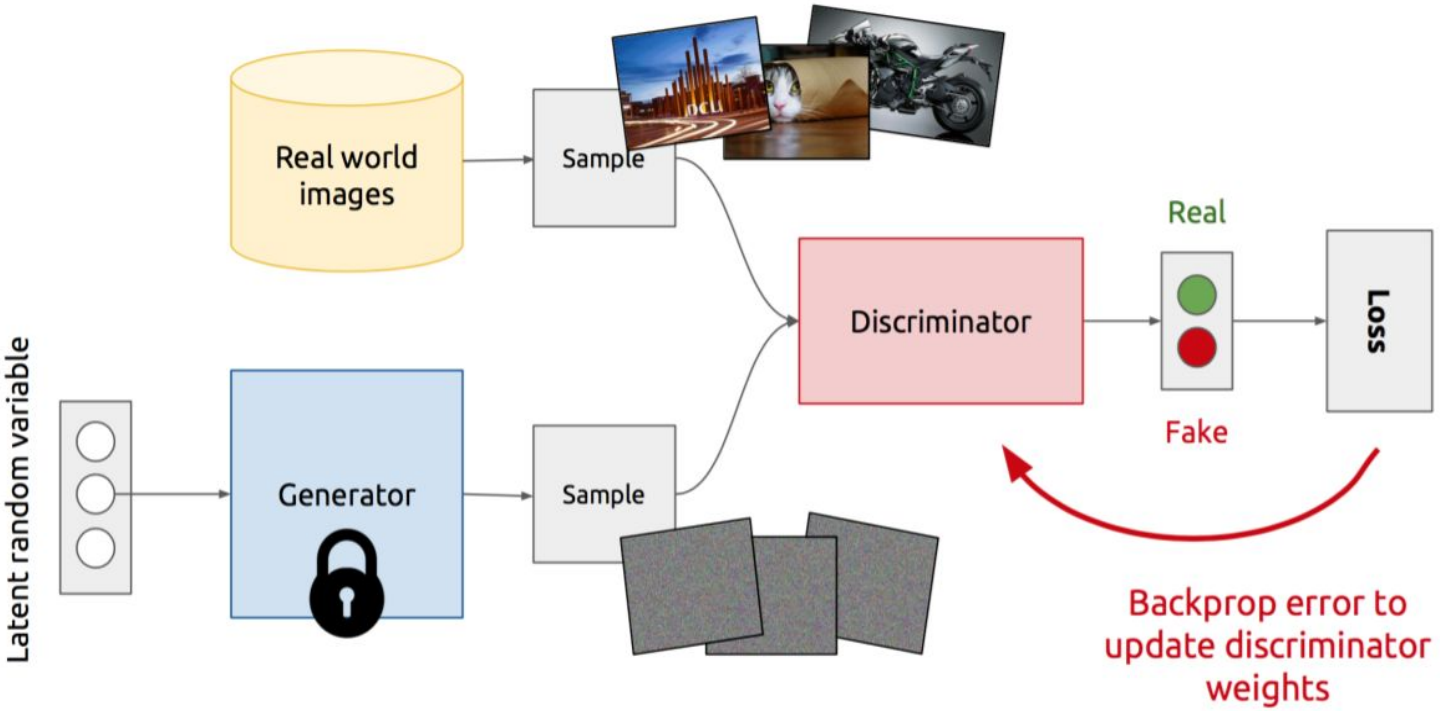


- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

Training Generator



Training Discriminator



Why Generative Models?

- We've only seen discriminative models so far
- Given an image X, predict a label Y
- Estimates P(Y|X)

- Discriminative models have several key limitations
- Can't model P(X), i.e. the probability of seeing a certain image
- Thus, can't sample from P(X), i.e. can't generate new images

- Generative models (in general) cope with all of above
- Can model P(X)
- Can generate new images

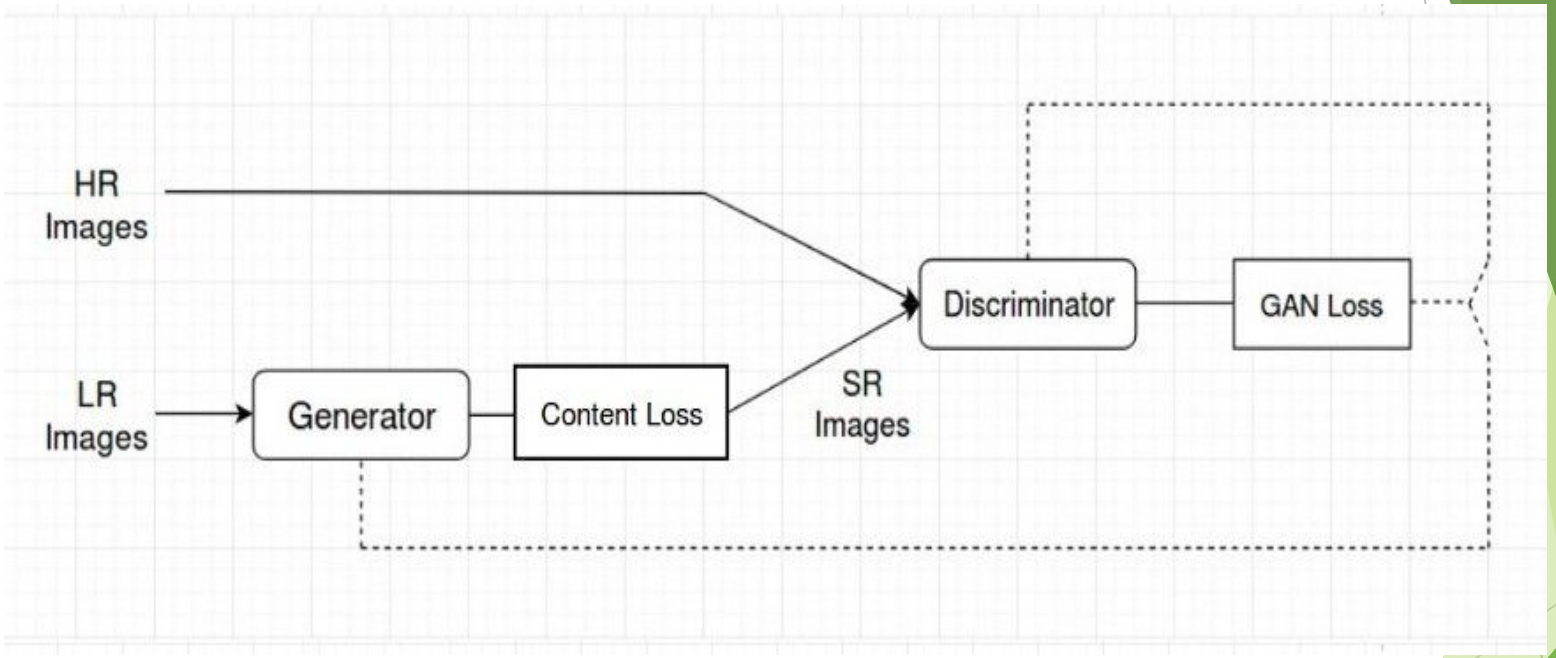
SRGAN

Idea behind SRGAN :

- The problem behind other ways for Single image super resolution is that how can we recover finer texture details from low resolution image so that image is not distorted.
- Our work has largely focused on minimizing the mean squared reconstruction error.
- The results have high peak signal-to-noise ratios(PSNR), that explains we have good image quality results.
- But they are often lacking high-frequency details and are perceptually unsatisfying as they are not able to match the fidelity expected in high resolution images.

- So we need a stable model which can capture the perceptual differences between the model's output and the ground truth image.
- To achieve this we will use Perceptual loss function which comprise of Content and Adversarial loss. Other than that SRGAN uses residual blocks to it's construction..

SRGAN Architecture



Super-resolution GAN applies a deep network in combination with an adversary network to produce higher resolution images.

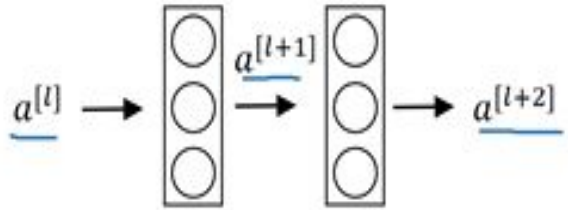
Training procedure

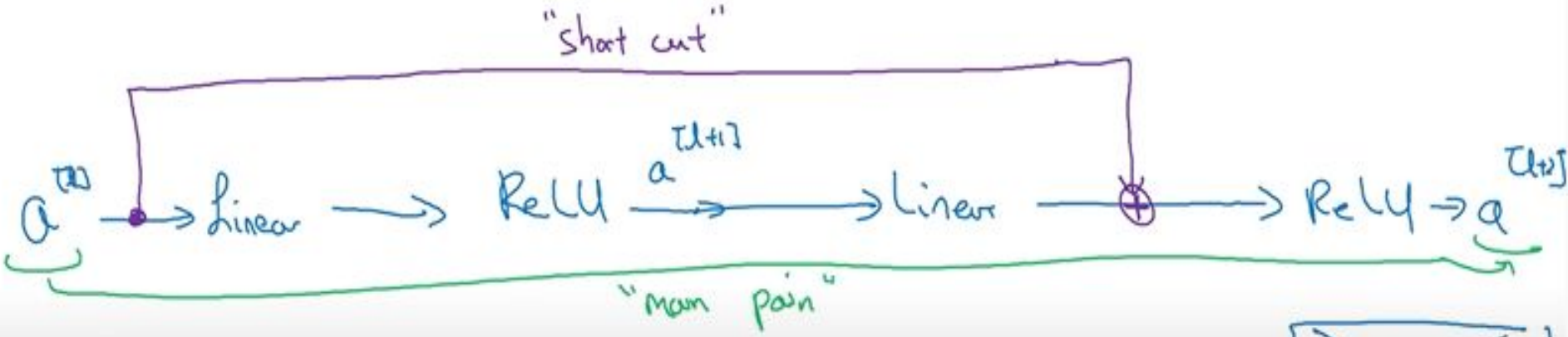
- We process the HR(High Resolution) images to get down-sampled LR(Low Resolution) images. Now we have both HR and LR images for training data set.
- We pass LR images through Generator which up-samples and gives SR(Super Resolution) images.
- We use a discriminator to distinguish the HR images and back-propagate the GAN loss to train the discriminator and the generator.

Residual Networks:

- In Recent years, we have seen tremendous growth in the field of Image Processing and Recognition
- Deep Neural Networks are becoming increasingly deeper and more complex.
- Adding More Layers can make Models more Robust but beyond a point, the Training Accuracy Starts Reducing.
- But adding many layers is helpful in extracting important features from complex images.
- Hence, we use Residual Networks (ResNet)

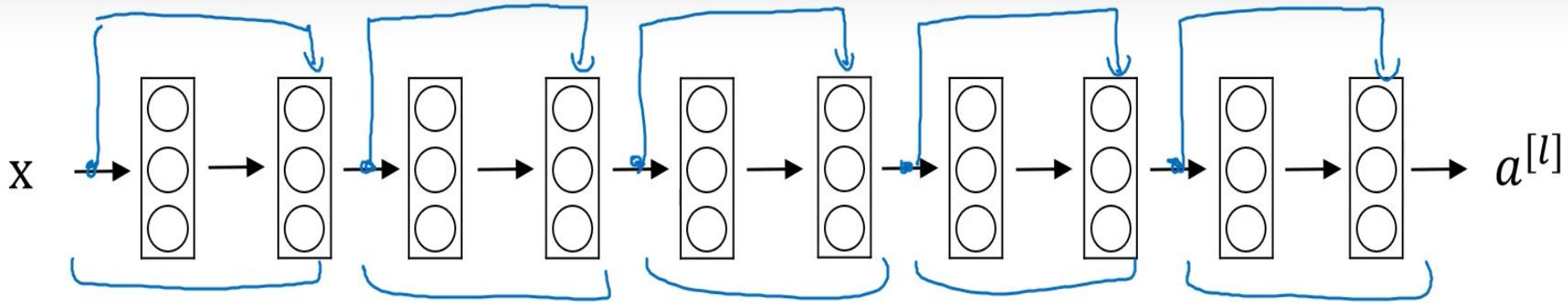
Residual block

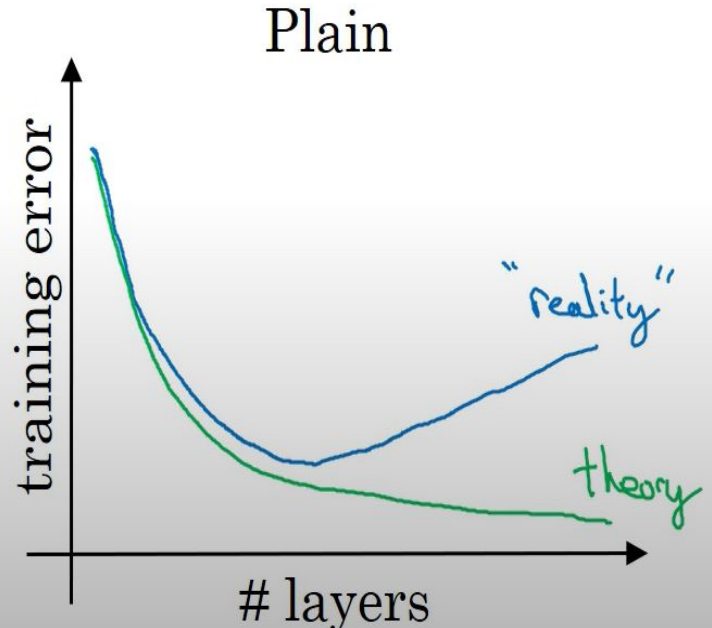


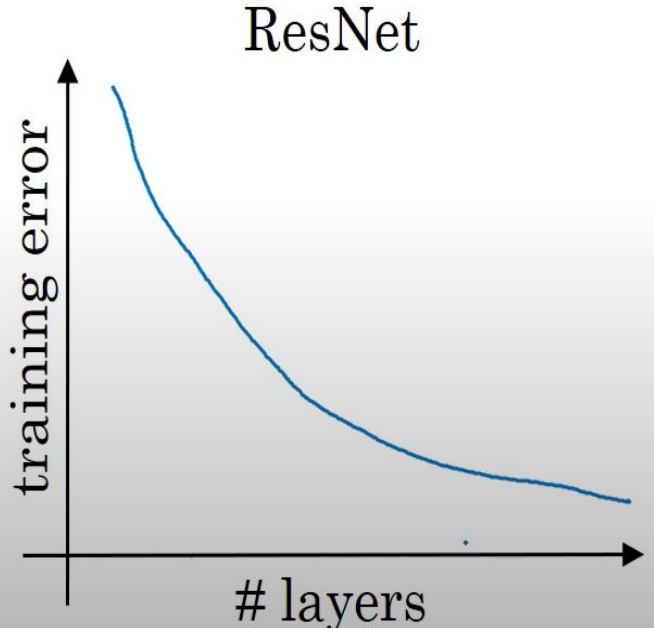


$$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]}) \qquad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g(z^{[l+2]})$$

$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right)$$


x
$a^{[l]}$

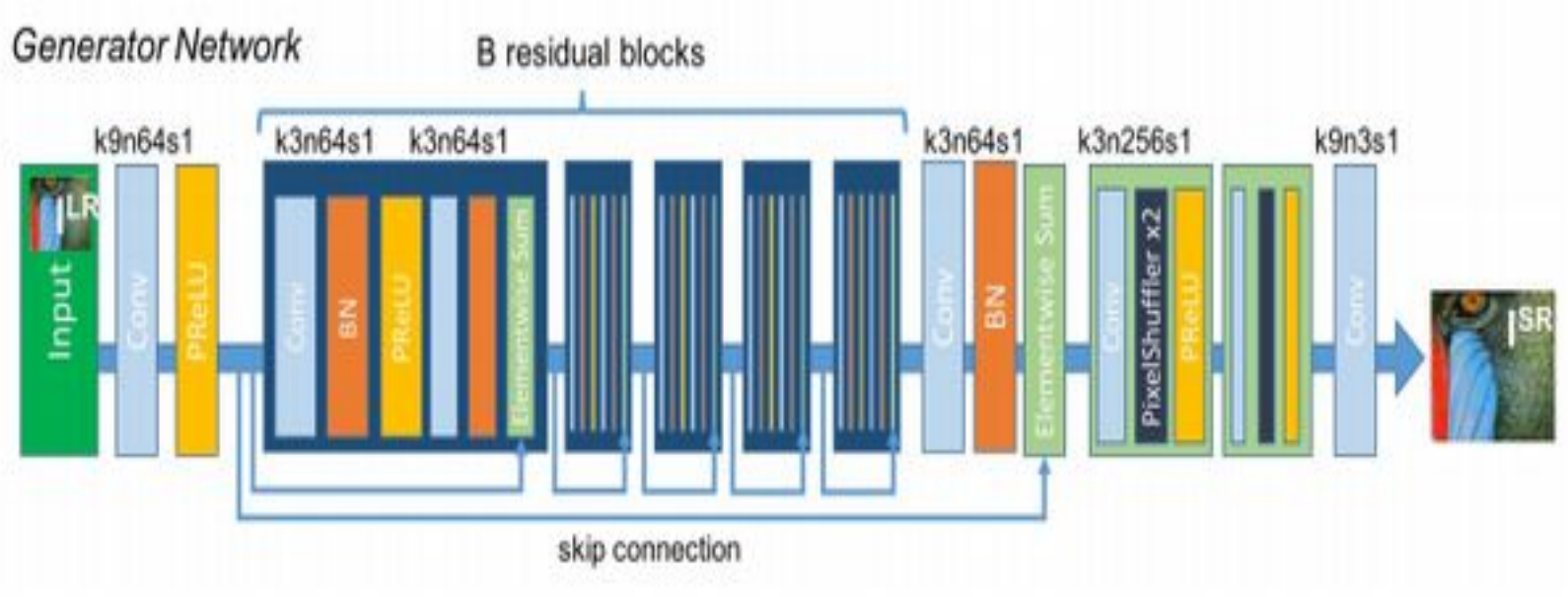

Plain
training error
"reality"
theory
layers


ResNet
training error
layers

Generator Architecture

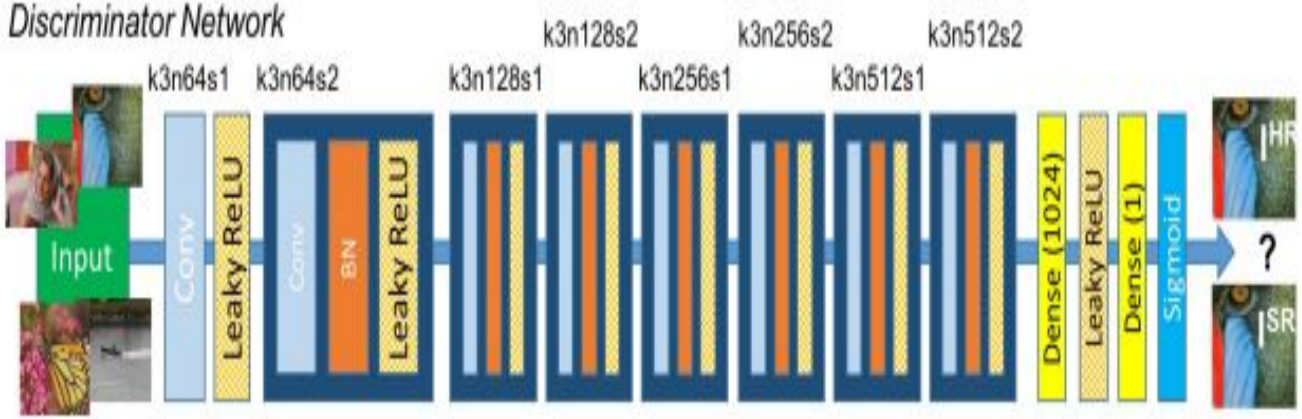
- The generator architecture contains residual network instead of deep convolution networks because residual networks are easy to train and allows them to be substantially deeper in order to generate better results.
- There are B residual blocks (16), originated by ResNet. Within the residual block, two convolutional layers are used, with small 3×3 kernels and 64 feature maps followed by batch-normalization layers and ParametricReLU as the activation function.
- The resolution of the input image is increased with two trained sub-pixel convolution layers.

- This generator architecture also uses parametric ReLU as an activation function which instead of using a fixed value for a parameter of the rectifier (alpha) like LeakyReLU. It adaptively learns the parameters of rectifier and improves the accuracy at negligible extra computational cost.
- During the training, A high-resolution image (HR) is downsampled to a low-resolution image (LR). The generator architecture than tries to upsample the image from low resolution to super-resolution. After then the image is passed into the discriminator, the discriminator and tries to distinguish between a super-resolution and High-Resolution image and generate the adversarial loss which then backpropagated into the generator architecture.


Generator Network
B residual blocks
k9n64s1
k3n64s1
k3n64s1
k3n64s1
k3n256s1
k9n3s1
Input
LR
Conv
PReLU
Conv
BN
PReLU
Element-wise Sum
Conv
BN
Element-wise Sum
Conv
PixelShuffler x2
PReLU
Conv
SR
skip connection

Discriminator Architecture

- The task of the discriminator is to discriminate between real HR images and generated SR images.
- The network contains eight convolutional layers with of 3×3 filter kernels, increasing by a factor of 2 from 64 to 512 kernels. Strided convolutions are used to reduce the image resolution each time the number of features is doubled.
- The resulting 512 feature maps are followed by two dense layers and a leakyReLU applied between and a final sigmoid activation function to obtain a probability for sample classification.


Discriminator Network
k3n64s1
k3n64s2
k3n128s1
k3n128s2
k3n256s1
k3n256s2
k3n512s1
k3n512s2
Input
Conv
Leaky ReLU
Conv
BN
Leaky ReLU
Dense (1024)
Leaky ReLU
Dense (1)
Sigmoid
?
HR
SR

Loss Function

- The SRGAN uses perceptual loss function (LSR) which is the weighted sum of two loss components : content loss and adversarial loss.
- This loss is very important for the performance of the generator architecture

Adversarial loss :

This encourages our network to favor solutions that reside on the manifold of natural images, by trying to fool the discriminator network.

The generative loss l SR Gen is defined based on the probabilities of the discriminator DθD (GθG (I LR)) over all training samples as:

$$l_{Gen}^{SR} = \sum^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

ILR = Low Resolution Input Image
Here, DθD (GθG (ILR)) is the probability that the reconstructed image GθG (ILR) is a natural HR image

Content Loss:

Pixelwise MSE loss for the SRResnet architecture, which is most common MSE loss for image Super Resolution.

$$l_{MSE}^{SR} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}(I_{x,y}^{HR} - G_{\theta_G}(I^{LR})_{x,y})^2$$

- However MSE loss does not able to deal with high frequency content in the image that resulted in producing overly smooth images.

- Instead of relying on pixel-wise losses we use a loss function that is closer to perceptual similarity.
- We define the VGG loss based on the ReLU activation layers of the pre-trained 19 layer VGG network.

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

- With φi,j we indicate the feature map obtained by the j-th convolution (after activation) before the i-th max pooling layer within the VGG19 network.
- We then define the VGG loss as the euclidean distance between the feature representations of a reconstructed image GθG(ILR) and the reference image IHR.

Perceptual Loss

The definition of our perceptual loss function l SR is critical for the performance of our generator network

$$l^{SR} = \underbrace{\underbrace{l_{\mathrm{X}}^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$

Resolution after 1 Epoch

Obtained from Training Set

Resolution after 500 Epochs

Obtained from Training Set

Resolution after 1000 Epochs

Obtained from Training Set

Resolution after 1500 Epochs

Obtained from Training Set

Resolution after 1750 Epochs

Obtained from Training Set

Final output on an Image that is from Outside the Dataset (with Noise 0.1)

Final output on an Image that is from Outside the Dataset (with Noise 0.2)

Final output on an Image that is from Outside the Dataset (with Noise 0.4)

Original Denoised

Generated

Final output on an Image that is from Outside the Dataset (with Noise 0.5)

Original Denoised

Generated

Conclusion

- Hence in the Last Semester, we have worked on a model that removes noise from the input image and outputs an image that is perceptually blur.
- In this Semester, we have worked on a model that Increases the Perceptual Quality of the Denoised Images.

Thank You